A

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE

UTILITY PATENT APPLICATION TRANSMITTAL LETTER UNDER 37 C.F.R. 1.53(b)

ATTORNEY DOCKET NO.:
2885/3A

Address to:
Assistant Commissioner for Patents
Washington D.C. 20231
Box Patent Application

Transmitted herewith for filing is the patent application of

Inventor(s): **Martin VORBACH**

For : **DATA PROCESSING SYSTEM**

Enclosed are:

1. **29** sheets of specification, **4** sheets of claims, and **1** sheet of abstract

2. **11** sheets of drawings.

3. German Language Application including **24** sheets of specification, **3** sheets of claims, **1** sheet of abstract.

4. Declaration and Power of Attorney (copies from prior application (37 CFR 1.63(d)) (See paragraph 4 below).

5. Incorporation by Reference. The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under paragraph 3 above is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. Continuing application information:

   This application is a continuation of prior application Serial No. 08/544,435, filed November 17, 1995.

7. Other enclosures:

   a) Return Receipt Postcard
   b) Preliminary Amendment

8. The filing fee has been calculated as shown below, after entry of the accompanying Preliminary Amendment:

	NUMBER FILED	NUMBER EXTRA*	RATE ($)	FEE ($)
BASIC FEE				760.00
TOTAL CLAIMS	64 - 20 =	44	18 00	792.00
INDEPENDENT CLAIMS	10 - 3 =	7	78 00	546.00
MULTIPLE DEPENDENT CLAIM PRESENT			260.00	--
*Number extra must be zero or larger			TOTAL	2,098.00
If applicant is a small entity under 37 C.F.R. §§ 1.9 and 1.27, then divide total fee by 2, and enter amount here.			SMALL ENTITY TOTAL	1,049.00

8. Small entity statement was filed in prior application. Small entity status is still proper and desired.

9 Please charge the required application filing fee of **$1049.00** to the deposit account of **Kenyon & Kenyon**, deposit account number **11–0600**. Small entity status was established in the parent application and is still deemed to be proper.

10. The Commissioner is hereby authorized to charge payment of the following fees, associated with this communication or arising during the pendency of this application, or to credit any overpayment to deposit account number **11-0600**:

A. Any additional filing fees required under 37 C.F.R. § 1.16;

B. Any additional patent application processing fees under 37 C.F.R. § 1.17;

C. Any additional patent issue fees under 37 C.F.R. § 1.18;

D. Any additional document supply fees under 37 C.F.R. § 1.19;

E. Any additional post-patent processing fees under 37 C.F.R. § 1.20; or

F. Any additional miscellaneous fees under 37 C.F.R. § 1.21.

11. A copy of this sheet is enclosed.

Dated: 12 April 1999

By: ______________________
Michelle M. Carniaux (Reg. No. 36,098)

KENYON & KENYON
One Broadway
New York, New York 10004
U.S.A.
(212) 425-7200 (phone)
(212) 425-5288 (facsimile)

#4

APPLICANT OR PATENTEE: Martin Vorbach

SERIAL NO. OR PATENT NO. 08/544,435 Attorney's Docket No.: ZAH0021

FILED OR ISSUED: November 17, 1995

TITLE: DATA PROCESSING SYSTEM

VERIFIED STATEMENT (DECLARATION) CLAIMING SMALL ENTITY STATUS
(37 CFR 1.9(f) and 1.27(b)) - INDEPENDENT INVENTOR

As a below named inventor, I hereby declare that I qualify as an independent inventor as defined in 37 CFR 1.9(c) for purposes of paying reduced fees under section 41(a) and (b) of Title 35, United States Code, to the Patent and Trademark Office with regard to the invention entitled: DATA PROCESSING SYSTEM described in:

[ ] The specification filed herewith

[X] Application Serial No. 08/544,435, filed November 17, 1995.

[ ] Patent No. ____________, issued ____________.

I have not assigned, granted, conveyed or licensed and am under no obligation under contract or law to assign, grant, convey or license, any rights in the invention to any person who could not be classified as an independent inventor under 37 CFR 1.9(c) if that person had made the invention, or to any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

Each person, concern or organization to which I have assigned, granted, conveyed, or licensed or am under an obligation under contract or law to assign, grant, convey, or license any rights in the invention is listed below:

[X] no such person, concern, or organization

[ ] persons, concerns, or organizations listed below *

* NOTE: Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities (37 CFR 1.27).

FULL NAME: ____________
ADDRESS: ____________

[ ] INDIVIDUAL [ ] SMALL BUSINESS CONCERN [ ] NONPROFIT ORGANIZATION

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate (37 CFR 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Martin Vorbach NAME OF INVENTOR	NAME OF INVENTOR	NAME OF INVENTOR
X Martin Vorbach SIGNATURE OF INVENTOR	SIGNATURE OF INVENTOR	SIGNATURE OF INVENTOR
X 17.01.96 DATE	DATE	DATE

EL234414395US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s) : Martin VORBACH et al.

Serial No. : To Be Assigned

Filed : Herewith

For : DATA PROCESSING SYSTEM

Examiner : To Be Assigned

Art Unit : To Be Assigned

Assistant Commissioner
  for Patents
Washington, D.C. 20231

PRELIMINARY AMENDMENT

SIR:

Kindly amend the above-identified application before examination, as set forth below.

IN THE CLAIMS:

Please cancel claims 1-14, without prejudice.

Please add the following new claims:

15. (New) A massively parallel data processing apparatus comprising:

a plurality of computing cells arranged in a multidimensional matrix, the plurality of computing cells capable of simultaneously manipulating a plurality of data, each of the plurality of computing cells including:

an input interface for receiving a plurality of input signals;

a plurality of logic members, at least one of the plurality of logic members coupled to

Express Mail No EL234414395US

the input interface, the plurality of logic members processing the plurality of input signals;

at least one coupling unit selectively coupling at least one of the plurality of logic members to another of the plurality of logic members a function of at least one of a plurality of configuration signals;

a register unit selectively storing a portion of the processed input signals; and

an output interface for transmitting the processed input signals;

wherein the input interface of at least one of the plurality of computing cells is selectively coupled to the output interface of at least another of the plurality of computing cells; and

a configuration interface for transmitting the plurality of configuration signals to at least some of the plurality of computing cells to configure the at least some of the plurality of computing cells.

16. (New) The massively parallel data processing apparatus according to claim 15, wherein the at least one coupling unit includes a multiplexer.

17. (New) The massively parallel data processing apparatus according to claim 15, further comprising:

a plurality of lines for selectively coupling at least one of the plurality of computing cells to another of the plurality of computing cells.

18. (New) The massively parallel data processing apparatus according to claim 15, further comprising:

a plurality of lines, at least one of the plurality of lines selectively coupling at least one of the plurality of computing cells to an adjacent one of the plurality of computing cells, at least another of the plurality of lines selectively coupling at least one of the plurality of computing cells to a non-adjacent one of the plurality of computing cells.

19. (New) The massively parallel data processing apparatus according to claim 15, further comprising:

a synchronization circuit providing a plurality of synchronization signals for synchronizing the configuration of the at least some of the plurality of computing cells.

20. (New) The massively parallel data processing apparatus according to claim 19, wherein the synchronization circuit includes at least one of the plurality of computing cells.

21. (New) The massively parallel data processing apparatus according to claim 15, further comprising

a configuration unit coupled to the configuration interface, the configuration unit generating the plurality of configuration signals, the at least some of the plurality of computing cells being configured as a function of the at least one configuration signal during operation of the massively parallel data processing apparatus such that others of the plurality of computing cells not being configured are not haltered or impaired in their operations.

22. (New) The massively parallel data processing apparatus according to claim 21, further comprising:

a configuration memory coupled to the configuration unit adapted to store the plurality of configuration signals.

23. (New) The massively parallel data processing apparatus according to claim 22, wherein the plurality of configuration signals are stored in the form of a plurality of configuration words.

24. (New) The massively parallel data processing apparatus according to claim 23, wherein the configuration unit manages a plurality of configuration programs, at least one of the configuration programs including at least one of the configuration words.

25. (New) The massively parallel data processing apparatus according to claim 21, wherein the configuration unit is controlled as a function of the plurality of synchronization signals.

26. (New) The massively parallel data processing apparatus according to claim 15, further comprising:

a configuration unit coupled to the configuration interface, the configuration unit adapted to dynamically reconfigure the massively parallel data processing apparatus.

27. (New) The massively parallel data processing apparatus according to claim 26, wherein the configuration unit is adapted to dynamically reconfigure the massively parallel data processing apparatus during a program sequence while data is still processed by the massively parallel data processing apparatus.

28. (New) The massively parallel data processing apparatus according to claim 27, wherein said configuration unit is adapted to dynamically reconfigure without influencing the data to be processed.

29. (New) The massively parallel data processing apparatus according to claim 15, wherein the input interface of at least

one of the plurality of computing cells is coupled to an external memory device.

30. (New) The massively parallel data processing apparatus according to claim 15, wherein the input interface of at least one of the plurality of computing cells is coupled to a peripheral device.

31. (New) The massively parallel data processing apparatus according to claim 15, wherein the input interface of at least one of the plurality of computing cells is coupled to at least a second massively parallel data processing apparatus.

32. (New) The massively parallel data processing apparatus according to Claim 15, further comprising:

a plurality of lines, at least one of the plurality of lines selectively coupling at least one of the plurality of computing cells to an adjacent one of the plurality of computing cells, at least another of the plurality of lines selectively coupling at least one of the plurality of computing cells to a non-adjacent one of the plurality of computing cells;

wherein the at least another of the plurality of lines are divided in a plurality of segments, wherein each of the plurality of segments is connected to at least another one of the plurality of segments by a tristate-bus-driver.

33. (New) The massively parallel data processing apparatus according to claim 15, further comprising:

a plurality of lines, at least one of the plurality of lines selectively coupling at least one of the plurality of computing cells to an adjacent one of the plurality of computing cells, at least another of the plurality of lines selectively coupling at least one of the plurality of computing cells to a non-adjacent one of the plurality of computing cells;

wherein the at least another of the plurality of lines are divided in a plurality of segments, wherein each of the plurality of segments is connected to at least another one of the plurality of segments by a transmission gate.

34. (New) The massively parallel data processing apparatus according to claim 15, further comprising:

a plurality of lines, at least one of the plurality of lines selectively coupling at least one of the plurality of computing cells to an adjacent one of the plurality of computing cells, at least another of the plurality of lines selectively coupling at least one of the plurality of computing cells to a non-adjacent one of the plurality of computing cells;

wherein the said at least another of the plurality of lines are divided in a plurality of segments, wherein each of the plurality of segments is connected to at least another one of the plurality of segments by an electrical switch.

35. (New) A massively parallel data processing apparatus, comprising:

a plurality of computing cells arranged in a multidimensional matrix, each of the plurality of computing cells being configurable and reconfigurable, each of the plurality of computing cells capable of processing a first plurality of data words simultaneously with the processing of a second plurality of data words by others of the plurality of computing cells, wherein each of the plurality of computing cells is configured by a first set of configuration words, and wherein each of the plurality of computing cells is reconfigured by a second set of configuration words; and a plurality of buses, wherein each of the plurality of computing cells are connectable to at least one of the plurality of computing cells using at least one of the plurality of buses.

36. (New) The massively parallel data processing apparatus of claim 35, wherein each of the plurality of computing cells maintaining a first configuration for a first period of time.

37. (New) The massively parallel data processing apparatus of claim 36, wherein at least some of the plurality of computing cells are capable of detecting a need for a reconfiguration.

38. (New) The massively parallel data processing apparatus of claim 36, further comprising:

at least one memory coupled to the processing apparatus for storing at least one of multiple data and processing results.

39. (New) The massively parallel data processing apparatus of claim 36, further comprising:

at least one interface for transferring data coupled to at least one of an external computer and a memory device.

40. (New) The massively parallel data processing apparatus of claim 36, further comprising:

at least one interface coupled to at least one peripheral device.

41. (New) The massively parallel data processing apparatus of claim 36, further comprising:

an interface structure optimized for coupling a plurality of the massively parallel data processing apparatuses.

42. (New) The massively parallel data processing apparatus of claim 36, further comprising:

a compiler capable of reconfiguring each of the plurality of computing cells independently of others of the plurality of computing cells such that portions of the massively parallel data processing apparatus not being reconfigured are not impaired in their operation.

43. (New) The massively parallel data processing apparatus of claim 42, further comprising:

a memory, coupled to the compiler, for storing the first set of configuration words and the second set of configuration words.

44. (New) The massively parallel data processing apparatus of claim 42, wherein the compiler manages a plurality of configuration programs.

45. (New) The massively parallel data processing apparatus of claim 35, further comprising:

a synchronization circuit providing a plurality of synchronization signals for synchronizing the configuration of the at least some of the plurality of computing cells.

46. (New) The massively parallel data processing apparatus of claim 45 wherein at least one of the plurality of computing cells provides status information to the synchronization circuit.

47. (New) A massively parallel data processing apparatus, comprising:

a programmable logic device, the programmable logic device including a plurality of logic elements arranged in a multidimensional matrix, each of the plurality of logic elements being configurable and reconfigurable, each of the plurality of logic elements capable of processing a first plurality of binary signals simultaneously with the processing of a second plurality of binary signals by others of the plurality of logic units, wherein each of the plurality of logic elements is configured by a first set of configuration words, and wherein each of the plurality of logic elements is reconfigured by a second set of configuration words, the programmable logic device further comprising a plurality of buses, wherein each of the plurality of logic elements are connectable to at least one of the plurality of logic elements using at least one of the plurality of buses;

at least one memory device coupled to the programmable logic device for storing at least one of i) multiple data to be processed by the programmable logic device, and ii) processing results of the programmable logic device; and

at least one of i) an interface coupled to at least one external computer and ii) a further memory device coupled to the at least one external computer, for transferring at least one of multiple data to be processed by the programmable logic device and processing results of the programmable logic device.

48. (New) The massively parallel data processing apparatus of claim 47, wherein each of the plurality of logic elements maintaining a first configuration for a first period of time.

49. (New) The massively parallel data processing apparatus of claim 48, wherein at least some of the plurality of logic elements are capable of detecting a need for a reconfiguration.

50. (New) The massively parallel data processing apparatus of claim 47, further comprising:

at least one peripheral interface coupled to at least one peripheral device.

51. (New) The massively parallel data processing apparatus of claim 47, further comprising:

an interface structure optimized for coupling a plurality of the massively parallel data processing apparatuses.

52. (New) The massively parallel data processing apparatus of claim 47, further comprising:

a compiler capable of reconfiguring each of the plurality of logic elements independently of others of the plurality of logic elements such that portions of the massively parallel data processing apparatus not being reconfigured are not impaired in their operation.

53. (New) The massively parallel data processing apparatus of claim 52, further comprising:

a memory, coupled to the compiler, for storing the first set of configuration words and the second set of configuration words.

54. (New) The massively parallel data processing apparatus of claim 52, wherein the compiler manages a plurality of configuration programs.

55. (New) The massively parallel data processing apparatus of claim 47, further comprising:

a synchronization circuit providing a plurality of synchronization signals for synchronizing the configuration of the at least some of the plurality of computing cells.

56. (New) The massively parallel data processing apparatus of claim 51 wherein at least one of the plurality of logic

elements provides status information to the synchronization circuit.

57. (New) A data processor, comprising:

cells arranged in a multi-dimensional pattern;

a first compiler for individually accessing and individually configuring at least some of the cells, the first compiler selectively grouping the at least some of the cells with neighboring cells into functional units to perform a first function, the first compiler further selectively regrouping selected ones of the at least some of the cells into different functional units to perform a second function different than the first function while simultaneously others of the at least some of the cells process data.

58. (New) The data processor according to claim 57, wherein the first compiler receives state information from at least one cell in each functional unit and selectively regroups the selected ones of the at least some of the cells as a function of the state information.

59. (New) The data processor according to claim 57, further comprising:

a priority decoder for determining an order for the first compiler to selectively regroup the selected ones of the at least some of the cells.

60. (New) The data processor according to claim 57 wherein the first compiler selectively regroups the selected one of the at least some of the cells by transmitting configuration data to at least one of the at least some of the cells, the at least one of the at least some of the cells reconfiguring to logically coupled to another of the cells as a function of the configuration data.

61. (New) The data processor according to claim 57, further comprising:

second cells arranged in a second multi-dimensional pattern coupled in cascade with the cells.

62. (New) A data processor, comprising:

cells arranged in a multi-dimensional pattern, at least one of the cells being selectively coupled to a first one of the cells to form a first functional unit at a first time, the first functional unit performing a first function, the at least one of the cells capable of being regrouped with a second one of the cells to form a second functional unit at a second time different from the first time, the second functional unit performing a second function different from the first function, the at least one of the cells regrouping as a function of reconfiguration data; and

a first compiler receiving state information regarding the state of the first functional unit and transmitting reconfiguration data to the at least one of the cells as a function of the received state information.

63. (New) The data processor according to claim 62, wherein the first compiler determines a program flow as a function of the state information.

64. (New) The data processor according to claim 62, wherein selected ones of the at least one of the cells reconfigure to perform the respective second function while simultaneously others of the at least some of the cells process data.

65. (New) The data processor according to claim 62, further comprising:

a priority decoder determining an order for the first compiler to transmit the reconfiguration data.

66. (New) The data processor according to claim 62, further comprising:

a segmented bus selectively coupling each of the cells to others of the cells.

67. (New) The data processor according to claim 66, wherein the segmented bus including a first segment and a second segment, the first segment providing communication between a first one of the cells and a second one of the cells, the second segment providing communication between a third one of the cells and a fourth one of the cells.

68. (New) The data processor according to claim 62, further comprising:

second cells arranged in a second multi-dimensional pattern coupled in cascade with the cells.

69. (New) A method for configuring a data processor, the data processor including cells arranged in a multi-dimensional pattern, comprising the steps of:

grouping at least some of the cells into functional units;

processing data by the functional units;

receiving, by a reconfiguration unit, state information regarding at least one of the functional units;

transmitting, by the reconfiguration unit, respective configuration data to the at least one of the functional units as a function of the received state information; and

reconfiguring at least one of the cells in the at least one of the functional units as a function of the respective configuration data while simultaneously other functional units continue processing data.

70. (New) The method according to claim 69, further comprising the step of:

regrouping the at least one of the cells with another of the cells into a different functional unit.

71. (New) The method according to claim 69, wherein the receiving step includes the step of receiving the state information from the at least one of the functional units.

72. (New) A method for configuring a data processor, the data processor including cells arranged in a multi-dimensional pattern, comprising the steps of:

individually configuring at least some of the cells to form functional units;

processing data by at least some of the functional units;

individually reconfiguring at least one cell of at least one of the functional units to form a different functional unit while simultaneously others of the functional units continue processing data.

73. (New) The method according to claim 72, further comprising the steps of:

receiving, by a reconfiguration unit, state information regarding at least one of the functional units;

transmitting, by the reconfiguration unit, reconfiguration data to the at least one cell, wherein the step of reconfiguring includes the step of reconfiguring, by the at least one cell, as a function of the reconfiguration data transmitted by the reconfiguration unit.

74. (New) A method for configuring a data processor, the data processor including cells arranged in a multi-dimensional pattern, comprising the steps of:

grouping at least some of the cells into functional units;

enabling by a state machine the at least some of the cells;

after the enabling step, processing data by the functional units;

receiving, by a reconfiguration unit, state information regarding at least one of the functional units;

transmitting, by the reconfiguration unit, respective configuration data to the at least one of the functional units as a function of the received state information;

disabling by the state machine at least some of the cells of the at least one of the function units while others of the function units are still enabled; and

after the disabling step, reconfiguring at least one of the cells in the at least one of the functional units as a function of the respective configuration data while simultaneously other functional units continue processing data.

75. (New) The method according to claim 74, wherein the enabling step includes the step of:

transmitting by the state machine first synchronization signals to the at least some of the cells.

76. (New) The method according to claim 75, wherein the disabling step includes the step of:

transmitting by the state machine second synchronization signals to the at least some of the cells.

77. (New) A method of processing data within a data processor, the data processor including cells arranged in a multi-dimensional pattern, comprising the steps of:

grouping at least some of the cells into functional units;

enabling by a state machine a transfer of data between a first plurality of the cells depending on the presence of the data and a state of at least some of the cells, the first plurality of the cells being in a READY state for at least one of i) accepting new data, and ii) transmitting a result; and

disabling by a state machine the transfer of the data depending on the presence of the data and the state of at least some of the cells, the at least some of the cells being in a NOT ready state for accepting the new data and for transmitting the results.

78. (New) A method for configuring a data processor, the data processor including cells arranged in a multi-dimensional pattern, comprising the steps of:

grouping at least some of the cells into functional units;

enabling by a state machine the at least some of the cells;

after the enabling step, processing data by the functional units;

receiving, by a reconfiguration unit, state information regarding at least one of the functional units;

disabling by the state machine at least some of the cells of the at least one of the function units while others of the function units are still enabled;

transmitting, by the reconfiguration unit, respective configuration data to the at least one of the functional units as a function of the received state information; and

after the disabling step, reconfiguring at least one of the cells in the at least one of the functional units as a function of the respective configuration data while simultaneously other functional units continue processing data.

REMARKS

Claims 1-14 have been canceled, without prejudice. Claims 15-78 have been added. No new matter has been added. It is respectfully submitted that the subject matter of the present application is new, non-obvious, and useful. Prompt consideration and allowance of the application are respectfully requested.

Respectfully submitted,
KENYON & KENYON

Dated: 12 April 1999 By:

Michelle Carniaux
Reg. No. 36,098

One Broadway
New York, N.Y. 10004
(212) 908-6036 (telephone)
(212) 425-5288 (facsimile)

77151-1

Martin VORBACH

Data Processing System

Background of the Invention

The present invention relates to a data processing system, i.e., a hardware unit for logical manipulation (linkage) of data (information) available in binary form.

Data processing systems of this type have by now been known long since, and they have already found broad application and recognition. The basic structure and operation of prior data processing systems can be defined approximately as comprising an arithmetic-logic linking unit in which the data to be linked are processed according to program instructions (software). In the process, the data are retrieved appropriately via a controller by more or less complex addressing procedures and, to begin with, kept ready in working registers. Following the logical linkage, the new data are then filed again in a preset memory location. The arithmetic-logic linkage unit consists of logical linking modules (gates, members) that are coupled to one another in such a way that the data to be manipulated allows in accordance with the underlying software logical processing using the four basic arithmetic operations.

It is easily perceivable that on the basis of the known structures there is relatively much computing time required for reading the data to be manipulated, transferring the data to the working registers, passing the data on to specific logic modules in the arithmetic-logic linking unit and, lastly, store the data again. Also, it is readily evident that the hardware structure of the arithmetic-logic linking unit cannot be considered as optimal inasmuch as the integrated logic hardware modules are actively used within the overall system always only in one and the same way. Similarly, a compilation of functions in so-called pipelines is very much impeded or restricted by strict hardware specification, which of necessity means frequent register reloading between working registers and central processing unit. Furthermore, such modules lend themselves poorly to cascading and require then substantial programming work.

An additional advantage of the present invention is that a widely scalable parallelity is available. Created here is a basis for a fast and flexible creation of neuronal structures such as to date can be simulated only at considerable expense.

The objective underlying the present invention is to propose a data processing system which hereafter will be referred to as data flow processor (DFP), where a greater, or better, flexibility of the overall structure and data flow as well as pipelining and cascading options result in increased processor capacity, or linking capacity.

Besides its employment as strictly a data flow processor, the DFP is meant to be able to handle the following further tasks:

- utilization as universal module in setting up conventional computers, making the structure simpler and less expensive;
- utilization in neuronal networks.

This objective is accomplished by providing an integrated circuit (chip) with a plurality of cells which, in particular, are arranged orthogonally to one another, each with a plurality of logically same and structurally identically arranged cells, whose arrangement and internal bus structure is extremely homogeneous so as to facilitate programming. Nonetheless, it is conceivable to accommodate within a data flow processor cells with different cell logics and cell structures in order to increase the capacity in that, for example, there exist for memory access other cells than for arithmetic operations. A certain specialization can be advantageous, especially for neuronal networks. Coordinated with the cells is a loading logic by way of which the cells, by themselves and facultatively grouped in so-called MACROs, are so programmed that, for one, selective logical functions but, for another, also the linking of cells among themselves can be realized widely. This is achieved in that for each individual cell a certain memory location is available in which the configuration data are filed. These data are used to switch multiplexers or transistors in the cell so as to guarantee the respective cell function (refer to Fig. 12).

Summary of the Invention

The core of the present invention consists in proposing a data flow processor of cellular structure whose cells allows reconfiguration in an arithmetic-logical sense, quasi randomly, via an external loading logic. Of extreme necessity is that the respective cells allow reconfiguration individually and without affecting the remaining cells or disabling the entire module. Thus, the data flow processor according to the present invention can be "programmed" as an adder in a first operating cycle and as a multiplier in a subsequent operating cycle, wherein the number of cells required for addition or multiplication may well be different and the placement of already loaded MACROs is upheld. It is incumbent upon the loading logic, or compiler, to partition the newly loaded MACRO within the available cells (i.e., to dissect said MACRO in such a way that it allows optimum insertion). The sequence control of the program is assumed by the compiler, which loads the appropriate MACROs in the module in accordance with the presently performed program section, the loading process being concomitantly controlled by the yet to be described synchronization logic, which determines the moment of reloading. Therefore, the DFP does not correspond to the known von Neumann architecture, since the data and program memories are separate. But this means at the same time increased security, since faulty programs cannot destroy a CODE, but merely DATA.

To give the data flow processor a workable structure, several cells — among others the input/output functions (I/O) and memory management functions — are loaded prior to loading the programs and remain constant usually for the entire run time. This is necessary in order to adapt the data flow processor to its hardware environs. The remaining cells are combined to so-called MACROs and allow reconfiguration, nearly at random and without affecting neighboring cells, during run time. To that end, the cells are individually and directly addressable.

The data flow processor may be adjusted, by way of the compiler, optimally and, as the case may be, dynamically to a task to be performed. Associated with this, e.g., is the great advantage that new standards or similar can be converted solely by reprogramming the data flow processor, not requiring as heretofore replacement and corresponding accrual of electronic scrap.

The data flow processors are suited for cascading, which results in a nearly random increase of parallelization, computing capacity as well as network size in neuronal networks. Especially important here is a clear homogeneous linkage of cells with the input/output pins (I/O pin) of the data flow processors, for maximum circumvention of program restrictions.

So-called shared memories can be used for better communication between data flow processors and the compiler. For example, programs from a hard disk situated in the O/I area of a data flow processor can be passed on to the compiler, in that the data flow processors write the data from the disk to the shared memory for retrieval by the compiler. This is especially important, since here, as already mentioned, not a von Neumann architecture exists, but a Harvard architecture. Similarly, shared memories are advantageous when constants defined in the program — which reside in the memory area of the compiler — are meant to be linked with data residing in the memory area of the data flow processors.

A special application of the inventional data flow processor is constituted in that — in conjunction with suitable input/output units, for one, and a memory for another — it may form the basis for a complete (complex) computer.

A further area of application is setting up large neuronal networks. Its particular advantage is constituted here by its high gate density, its excellent cascading as well as its homogeneity. A learning process comprising a change of individual axiomatic connections, or individual cell functions, is on customary modules just as difficult to perform as the setup of homogeneous and at the same time flexible cell structures. Dynamic reconfigurability enables for the time an optimum simulation of learning processes.

Brief Description of the Drawings

The present invention will be more fully explained hereafter with the aid of the drawings which show the following:

Fig. 1, a wiring symbol for an 8-bit adder;

Fig. 2, a wiring symbol for an 8-bit adder according to Fig. 1 consisting of eight 1-bit adders;

Fig. 3, a logical structure of a 1-bit adder according to Fig. 2;

Fig. 4, a cell structure of the 1-bit adder according to Fig. 3;

Fig. 5, an 8-bit adder composed according to the cell structure relative to Fig. 1;

Fig. 6, an unprogrammed SUBMACRO X consisting of four cells (analogous to a 1-bit adder relative to Fig. 4, or Fig. 5) with the necessary line connections;

Fig. 7, a partial section of an integrated circuit (chip) with a plurality of cells and a separate SUBMACRO X according to Fig. 6;

Fig. 8, an integrated circuit (chip) with an orthogonal structure of a quasi random plurality of cells and an externally assigned compiler;

Fig. 9, a first exemplary embodiment of a plurality of integrated circuits coupled to form a central processors (data flow processor) according to Fig. 8;

Fig. 10, a second exemplary embodiment of a plurality of integrated circuits coupled to form a central processor (data flow processor) according to Fig. 8;

Figs. 11a-11c an exemplary embodiment of a MACRO for adding two numerical series;

Fig. 12, an exemplary structure of a cell comprising multiplexers for selection of the respective logic modules;

Fig. 13, a synchronizing logic circuit using, e.g., a standard TTL module 74148;

Fig. 14, the cascading of four DFPs, with the connection between I/O pins shown only schematically (actually, a depicted connection means a plurality of lines);

Fig. 15, the homogeneity achieved by cascading;

Fig. 16a, the structure of the I/O cells, with the global connections not taken outside;

Fig. 16b, the structure of the I/O cells, but with the global connections taken outside;

Fig. 17a, the cascading resulting from Fig. 16a, depicting a corner cell and the two driver cells communicating with it, of the cascaded modules (compare with Fig. 14);

Fig. 17b, the cascading resulting from Fig. 16b, depicting a corner cell as well as the two driver cells communicating with it, of the cascaded modules (compare Fig. 14);

Fig. 18a, a multiplication circuit (compare Fig. 11a);

Fig. 18b, the internal structure of the DFP after loading (compare Fig. 11b);

Fig. 19a, a cascade circuit, with the adder of Fig. 11 and the multiplier of Fig. 18 wired in series for increased computing capacity;

Figs. 19b and 19c, the operating mode of the DFP in the memory, as well as the states of counters 47, 49;

Fig. 20, the block diagram of a conventional computer;

Fig. 21, the possible structure of the computer of Fig. 20 with the aid of an array of cascaded DFPs;

Fig. 22, a section of a DFP showing the line drivers.

Detailed Description

Fig. 1 illustrates a circuitry symbol for an 8-bit adder. Said wiring symbol consists of a quadratic module 1 with eight inputs A 0 ... 7 for a first data word A and eight inputs B 0 ... 7 for a second data word B (to be added). The eight inputs Ai, Bi are supplemented each by a further input Üein, by way of which, as the case may be, a carry-over is passed on to the module 1. In keeping with its function and purpose, the module 1 has eight outputs S 0 ... 7 for binary summands and a further output Üaus for any existing carry-over.

The circuitry symbol illustrated in Fig. 1 is shown in Fig. 2 as an arrangement of so-called SUBMACROs. Said SUBMACROs 2 consist each of a 1-bit adder 3 with one input each for the appropriate bits of the data word and a further input for a carry-over bit. Furthermore, the 1-bit adders 3 feature an output for the summand and an output for the carry-over Üaus.

Shown in Fig. 3 is the binary logic of a 1-bit adder, or SUBMACRO 2 according to Fig. 2. Analogous to Fig. 2, this logic features one input Ai, Bi each for the conjugate bits of the data to be linked; an input Üein is provided additionally for the carry-over. These bits are linked in accordance with the illustrated connections, or linkages, in two OR members 5 and three NAND members 6 so that on the output port Si and on the output for the carry-over Üaus the linkage results (Si, Üaus) corresponding to a full adder are present.

The invention sets in — as illustrated in Fig. 4 — where the implementation of SUBMACRO 2 shown in Fig. 3 or of one or several random functions in a suitable manner in a cell structure is involved. This is handled on the basis of logically and structurally identical cells 10, the individual logic modules of which are coupled with one another in accordance with the linkage function to be carried out, and at that, by means of the yet to be described compiler. According to the linking logic shown in Fig. 4 and deriving from the compiler according to Fig. 3 for a 1-bit adder, two cells 10.1, 10.2 each are with respect to the logic modules insofar identical as always an OR member 5 and a NAND member 6 are activated. The third cell 10.3 is used only as a line cell (PC conductor cell), and the fourth cell 10.4 is switched active with regard to the third NAND member 6. The SUBMACRO 2 consisting of the four cells 10.1 ... 10.4 is thus representative for a 1-bit adder, that is, a 1-bit adder of a data processing system according to the present invention is realized via four appropriately programmed (configured) cells 10.1 ... 10.4. (For the sake of completeness it is noted here that the individual cells possess an appreciably more extensive network of logic modules, or linkage members, and inverters, each of which can be switched active according to the relevant instruction by the compiler. Also provided, in addition to the logic modules, is a dense network of connecting lines between adjoining modules and for the setup of line and column bus structures for data transfer, so that appropriate programming on the part of the compiler allows implementation of quasi random logic linking structures.)

Fig. 5 illustrates for the sake of completeness the cell structure of an 8-bit adder in its entirety. The illustrated structure corresponds insofar to that relative to Fig. 2, with the 1-bit adders shown in Fig. 2 symbolically as SUBMACROs 2 being replaced each by a four-cell unit 10.1 ... 10.4. Based on the inventional data flow processor, this means that thirty-two cells of the available entirety of cells of a PC board fabricated cellularly with logically identical layout are accessed and configured, or programmed, by the compiler, in such a way that these thirty-two cells form one 8-bit adder.

In Fig. 5, a dash-dot bordering separates pictorially a SUBMACRO "X", which ultimately is to be viewed as a subunit comprised of four cells programmed to correspond to a 1-bit adder (10 according to Fig. 4).

1.The SUBMACRO "X" set off in Fig. 5 is illustrated in Fig. 6 as part of an integrated circuit (chip) 20, along with line and data connections. The SUBMACRO "X" consists of the four cells 10, which in accordance with the orthogonal structure possess per side four data connections (that is, totally 16 data connections per cell). The data connections join neighboring cells, so that it can be seen how, e.g., a data unit is being channeled from cell to cell. The cells 10 are activated, for one, via so-called local controls — which are local lines connected to all cells — and, for another, via so-called global lines, i.e., lines routed through the entire integrated circuit (chip) 20.

Fig. 7 illustrates a scaled-up section of an integrated circuit 20 that is allocated to an orthogonal raster of cells 10. As indicated in Fig. 7, e.g., a group of four cells 10 can thus be selected as SUBMACRO "X" and programmed, or configured, according to the 1-bit adder in Fig. 4.

A complete integrated circuit (chip) 20 is illustrated in Fig. 8. This integrated circuit 20 consists of a plurality of cells 10 arranged in the orthogonal raster and possesses on its outside edges an appropriate number of line connectors (pins) by way of which signals, specifically activation signals, and data can be fed and relayed. In Fig. 8, the SUBMACRO "X" according to Fig. 5-6 is distinguished again, and additionally there are further SUBMACROs separated and combined in subunits in accordance with specific functions and integrations. Coordinated with the integrated circuit (chip) 20, or superposed, is a compiler 30 by way of which the integrated circuit 20 is programmed and configured. The compiler 30 merely instructs the integrated circuit 20 how it should operate arithmetically-logically. With reference to Fig. 1 through 5, for one, Fig. 8 emphasizes the SUBMACRO "X" in Fig. 4 and Fig. 5; on the other hand, also a MACRO "Y" according to Fig. 1 and 2 is shown, which as a unit corresponds to an 8-bit adder.

Fig. 9 and 10 will respectively describe in the following a computer structure that bases on the integrated circuit 20 defined and illustrated above.

According to the first exemplary embodiment illustrated in Fig. 9, a plurality of integrated circuits 20 — analogous to the arrangement of the cells — are arranged in the orthogonal raster, with adjoining circuits being coupled, or linked, to one another via local bus lines 21. Consisting for instance of 16 integrated circuits 20, the computer structure features input/output lines I/O, by way of which the computer quasi communicates, i.e., transacts with the outside world. The computer according to Fig. 9 also features a memory 22, which according to the illustrated exemplary embodiment consists of two separate memories, each composed of RAM, ROM as well as a dual-ported RAM wired as shared memory to the compiler, which memories can be realized likewise as write-read memories or also as read memories only. Hierarchically coordinated with, or superimposed upon, the computer structure described so far is the compiler 30, by means of which the integrated circuits (data flow processor) 20 can be programmed and configured and linked.

The compiler 30 is based on a transputer 31, i.e., a processor with a microcoded set of instructions, with which transputer a memory 32 is coordinated. The connection between the transputer 31 and the data flow processor is based on an interface 33 for the so-called loading data, i.e., the data which program and configure the data flow processor for its task, and on an interface 34 for the previously mentioned computer memory 22, i.e., the shared memory.

The structure illustrated in Fig. 9 represents thus a complete computer that allows case-specific or task-specific programming and configuration via the compiler 30. For the sake of completeness it is noted yet that — as indicated by arrows in conjunction with the compiler 30 — several of these computers can be linked, i.e., coupled to one another.

A further exemplary embodiment of a computer structure is illustrated in Fig. 10. As opposed to Fig. 9, the local BUS lines between neighboring integrated circuits 20 are supplemented by central BUS lines 23 for solving, e.g., specific input or output problems. Also the memory 22 (shared memory) connects via central BUS lines 23 to the integrated circuits 20, and at that, as illustrated by groups of these integrated circuits. The computer structure illustrated in Fig. 10 features the same compiler 30 as illustrated with the aid of Fig. 9.

Fig. 11a serves to illustrate an adding circuit based on data flow processors according to the invention, basing on two numerical series An and Bn for all n's between 0 and 9. The task is forming the sum Ci = Ai + Bi, where the index i may assume the values 0 <=n < 9.

With reference to the illustration relative to Fig. 11a, the numerical series An is stored in the first memory RAM1, and at that, for instance starting with a memory address 1000h; the numerical series Bn is stored in a memory RAM2 at an address 0dfa0h; the sum Cn is written into RAM1, and at that, at the address 100ah.

Another counter 49 is connected, which merely increments the clock cycles authorized by the control circuit. This is to demonstrate in the following the reconfigurability of individual MACROs, without affecting those MACROs which do not participate in the reconfiguration.

Fig. 11a, to begin with, shows the actual addition circuit 40, which is comprised of a first register 41 for receiving the numerical series An and of a second register 42 for receiving the numerical series Bn. The two registers 41-42 are followed by an 8-bit adder corresponding to the MACRO1 illustrated in Fig. 1. The output of MACRO1 leads by way of a driver circuit 43 back to RAM1. The clock, or time, control of the addition circuit 40 is handled by a timer control (STATEMACHINE) 45 activated by a clock generator T and connected to the registers 41, 42 and the driver circuit 43.

The addition circuit 40 is functionally supplemented by an address circuit 46 for generating the address data for the addition results to be stored. The address circuit 46, in turn, consists of three MACROs 1 (according to Fig. 1) for address data generation, which MACROs 1 are switched as follows: the addresses to be linked, for An, Bn, Cn, are fed each via an input. These addresses are added to the output signals of a counter 47 and linked by the Statemachine 45 in such a way that a new target address prevails on the output. The task of counter 47 and comparator 48 is safeguarding that always the correct summands will be linked and that the process will abort always at the end of the numerical series, that is, at n = 9. Once the addition is completed, the timer control 45 generates a STOP signal, which switches the circuit to its passive side. Similarly, the STOP signal can be used as an input signal for a synchronizing circuit, in that the synchronizing logic can with the aid of this signal recognize that the overall function “adding” has been completed in accordance with the ML1 program described in the following, and the MACROs thus can be replaced by new ones (for example, STOP could be the signal Sync5).

The time sequence in the timer 45 (STATEMACHINE) may be represented as follows, in which context it is noted yet that a delay time T (in the form of clock cycles) is implemented in the timer control 45, between address generation and data receipt:

- In cycle 1, the counter 47 is incremented always by 1, while the comparator 48 tests whether n > 9 has been reached; in synchronism with these operations, the addresses for A, B, C are computed;
- in the cycle (T + 1), the summands A, B are read out and added;
- in cycle (T + 2), the sum C is stored.

In other words, this means that the operating loop and the addition proper require exactly (T + 2) clock cycles. T requires generally 2 ... 3 clock cycles, so that — as compared to conventional processors (CPU), which generally need 50 to several 100 clock cycles — a quite appreciable computing time reduction is possible.

The configuration presented with the aid of Fig. 11 shall be illustrated once more in the following by way of a hypothetical MACRO language ML1.

Given are the numerical series An and Bn

$\forall n: 0 <= n < 9$

To be formed are the sums Ci = Ai + Bi, with I ϵ N .

```
const n = 9;
array A [n] in RAM [1] at 1000h;
array B [n] in RAM [2] at 0dfa0h;
array C [n] in RAM [1] at 100ah;

for i = 0 to n with (A [i], B [i], C [i])
        Δ1;
        C = Δ 1 = A + B;
next;
```

RAM 1 is the 1st memory block

RAM 2 is the 2nd memory block

at follows the base address of the arrays

for is the loop start

next is the loop end

with ( ) follow the variables whose addresses are determined by the counting variable i

Δ T follows the delay time for a Statemachine in clock cycles.

Hence, the timing of the Statemachine looks as follows:

Cycle	Activity
1	Increment counter, compare to > 9 (yes => abort) and compute addresses for A, B, C
T + 1	Retrieve and add A, B
T + 2	Store in C

That is — as already mentioned — the loop and the addition require exactly one time T+2 clock cycles.

Fig. 11b shows the rough structure of the individual functions (MACROs) in a DFP. The MACROs are depicted in their approximate position and size and referenced using the appropriate numbers illustrated with the aid of Fig. 11a.

Fig. 11c shows the rough structure of the individual functions on the RAM blocks 1 and 2: the summands are successively read in ascending order from the RAM blocks 1 and 2 beginning with address 1000h, or 0dfa0h, and stored in RAM block 1 from address 100ah. Additionally, there are the counters 47 and 49, both counting during the sequence of the circuit from 0 to 9.

Following completion of the described program, a new program is to be loaded which handles the further processing of the results. The reloading is to take place at the run time. The program is given in the following:

Available are the numerical series An and Bn, with An given by the results Cn of the program run before:

$\forall n$:   0<=n<=9

To be formed are the products Ci = Ai * Bi with I ϵ N.

```
const n = 9;
array A [n] in RAM [1] at 100ah;
array B [n] in RAM [2] at 0dfa0h;
array C [n] in RAM [1] at 1015h;

for i = 0 to n with (A [i], B [i], C [i])
```

```
        Δ 1;
        C = Δ 1 = A * B;
    next;
```

The description of the individual instructions is already known, * symbolizes multiplication.

The MACRO structure is described in Fig. 18a, while Fig. 18b shows in known fashion the position and size of the individual MACROs on the chip; noteworthy is the size of the multiplier 2 as compared to the adder 1 of Fig. 11b. Fig. 18c demonstrates once again the effect of the function on the memory; counter 47 counts again from 0 to 9, that is, the counter is reset as the MACROs are reloaded.

Particular note is due counter 49. Assuming that reloading the MACROs takes 10 clock cycles, the counter 49 runs then from 9 to 19, since the module is being reloaded dynamically, that is, only the parts to be reloaded are halted while the rest continues to operate. As a consequence, the counter increments during the program sequence from 19 to 29. (This is to demonstrate the dynamic, independent reloading; in any prior module the counter would run again from 0 to 9, since it is reset).

Closer scrutiny of the problem gives rise to the question why not both operations, addition and multiplication, are carried out in one cycle, that is, the operation:

Available are the numerical series An and Bn, with An given by the result of Cn of the program run previously:

$\forall$ n: 0<=n<=9

To be formed are the products Ci = (Ai + Bi) * Bi with I $\in$ N.

```
path D;
const n = 9;
array A [n] in RAM [1] at 1000h;
array B [n] in RAM [2] at 0dfa0h;
array C [n] in RAM [1] at 100ah;

for i = 0 to n with (A [i], B [i], C [i])
        Δ 1;
        D = Δ 1 = AbB;
        C = Δ 1 = D * B;
next;
```

path D defines an internal double path that is not taken outside the DFP. Owing to an additional 1, the operation requires one clock cycle more than before, but is overall faster than the two above programs performed successively, because, for one, the loop is being passed only once and, for another, there is no reloading taking place.

Basically, the program formulation could also be as follows:

```
const n = 9;
array A [n] in RAM [1] at 1000h;
array B [n] in RAM [2] at 0dfa0h;
array C [n] in RAM [1] at 100ah;

for i = 0 to n with (A [i], B [i], C [i])
        Δ 1;
        C = Δ 2 =(A + B) * B;
next;
```

When the gate run times of the adder and the multiplier combined are smaller than one clock cycle, the operation (A+B)*B can be carried out also in one clock cycle, leading to a further appreciable speed increase:

```
const n = 9;
array A [n] in RAM [1] at 1000h;
array B [n] in RAM [2] at 0dfa0h;
array C [n] in RAM [1] at 100ah;

for i = 0 to n with (A [i], B [i], C [i])
        Δ 1;
        C = Δ 1 =(A + B) * B;
next;
```

A simple example of a cell structure is illustrated with the aid of Fig. 12. Cell 10 encompasses, e.g., an AND member 51, an OR member 52, an XOR member 53, an inverter 54 as well as a register cell 55. Furthermore, the cell 10 has on its input side two multiplexers 56, 57 with (corresponding to the sixteen inputs of the cell in Fig. 6), e.g., sixteen input connectors IN1, IN2 each. The (16:1) multiplexer 56/57 selects the data to be fed to the said logic members AND, OR, XOR 51 ... 53. These logic members are on the output end coupled to a (3:1) multiplexer 58 which, in turn, is coupled to the input of inverter 54, an input of register cell 55 and a further (3:16) multiplexer 59. The latter multiplexer 59 is hooked additionally to the output of inverter 54 and one output of register cell 55 and emits the output signal OUT.

For the sake of completeness it is noted that the register cell 55 is coupled to a reset input R and a clock input.

Superposed upon the illustrated cell structure, that is, of the cell 10, is now a compiler 30 that connects to the multiplexers 56, 57, 58 and 59 and activates them in accordance with the desired function.

For example, if the signals A2 are to be ANDed with B5, the multiplexers 56, 57 are switched active according to the lines "TWO" or "FIVE"; the summands proceed then to the AND member 51 and, with proper activation of multiplexers 58, 59, are surrendered at the output OUT. If a NAND linkage is to be carried out, e.g., the multiplexer 58 switches to the inverter 54, and the negated AND result prevails then on the output OUT.

To synchronize the restructuring (reloading) of the cells or MACROs with the compiler, a synchronization circuit that sends the appropriate signals to the compiler can be accommodated as a MACRO on the data flow processor — as required, since reloading is allowed only once the MACROs have finished their previous activity. This may necessitate a modification of the usual MACROs, since these must then make status information available to the synchronization circuit.

These status information usually signal the synchronization logic that some MACROs have finished their task, which from a programming aspect may mean, e.g., the termination of a procedure or reaching the termination criterion of a loop. That is, the program is continued at a different point, and the MACROs sending the status information can be reloaded. Besides, it may be of interest for the MACROs to be reloaded in a certain order. For that purpose, the individual synchronization signals may undergo a weighting by a priority decoder. Such — simple — logic is illustrated in Fig. 13. It possesses seven input signals by which the seven MACROs deliver their status information. In this case, 0 stands for "working" and 1 for "finished." The logic has three output signals that are transmitted to the compiler, with 000 designating rest status. With a signal present on one of the seven inputs, a decimal-binary conversion takes place. For example, Sync6 is represented as 110, indicating to the compiler that the MACRO servicing Sync6 has completed its task. With several synchronization signals present on the input simultaneously, the synchronization circuit transmits the signal with the highest priority to the compiler; for example, with Synch0, Sync4 and Sync6 present, the synchronization circuit first relays Sync6 to the compiler. Once the appropriate MACROs have been reloaded and Sync6 is no longer present, Sync4 is relayed etc. To demonstrate this principle, consideration may be given to the standard TTL module 74148.

The data flow processors are suited for cascading. Illustrated in Fig. 14, e.g., is the cascading of four DFPs. They have the appearance of a large homogeneous module (Fig. 15). Basically two cascading methods are conceivable with it:

a) Only the local connections between the cells are taken outside, meaning presently two I/O pins per edge cell and four I/O pins per corner cell.

However, the compiler/programmer must note that the global connections are not taken outside, for which reason the cascading is not completely homogeneous (global connections between several cells, usually between a complete cell row or cell column — refer to Fig. 6 — ; local connections exist only between two cells). Fig. 16a shows the structure of a DFP, Fig. 17a the resulting cascading of several DFPs (three being shown).

b) The local and global connections are taken outside, which drastically raises the number of required drivers-I/O pins and lines, in our example to six I/O pins per edge cell and twelve I/O pins per corner cell. The result is complete homogeneity in cascading.

Since the global connections, especially when using the cascading technique b), may become very long, the unpleasant effect may arise that the number of global connections is insufficient, since each connection — as known — can be used only by one signal. To minimize this effect, a driver may be looped through after a certain length of the global connections. For one, the driver serves to amplify the signal, which with long distances and correspondingly high loads is absolutely necessary while, for another, the driver may go into tristate and thus interrupt the signal. This allows the use of the sections left and right, or above and below the driver by different signals, provided the driver is in tristate, while otherwise a signal is looped through. Important is that the drivers of the individual global lines can be activated also individually, that is, a global signal may be interrupted, whereas the neighboring signal is looped through. Thus it is possible for different signals to be present sectionally on a global connection, whereas the global neighboring connection actually is used globally by one and the same signal (compare Fig. 22).

A special application of the inventional data flow processor is constituted in that — in conjunction with suitable input/output units, for one, and a memory for another — it may form the basis for a complete (complex) computer. A major part of the I/O functions may be implemented as MACROs on the data flow processor, and only special modules (Ethernet drivers, VRAMS ...) need momentarily be added externally. A change in specification, or improvements, involve then only — as already indicated — a software change of the MACRO; a hardware intervention is not required. It suggests itself here to specify an I/O connector by way of which the accessory modules can then be connected.

Fig. 20 shows the greatly simplified structure of a contemporary computer. Considerable parts can be saved by using a DFP module (Fig. 21), with the appropriate conventional modules (CPU, memory management, interfaces for SCSI, keyboard and video as well as those of the parallel and serial interfaces) being stored as MACROs in the cascaded DFPs. External wiring is required only for the parts that cannot be simulated by a DFP, such as memory and line drivers with non-TTL peaks or for high loads. Using DFPs makes for a favorable production, since one and the same module is used very frequently; the card layout is analogously simple, due to the homogeneous integration. Moreover, the structure of the computer is determined by the compiler, which here usually loads the DFP array only at the start of a run (after a reset), whereby a favorable error correction and extension option is given. Such computer notably can simulate several different computer structures by simply loading the structure of the computer to be simulated in the DFP array. It is noted that the DFP here is not working in its DFP function, but is merely available as a highly complex and freely programmable logic array while nonetheless differing from conventional modules by its especially good cascading ability.

What is claimed is:

Sub C1

~~1. A data processing system for manipulating data, comprising; an integrated circuit data flow processor, said data flow processor comprising a plurality of logically and structurally identical cells arranged orthogonal relative to one another, a plurality of connecting lines disposed between each said cells and a plurality of input and output ports, said cells connected to neighboring cells by a plurality of first data connections, said cells also connected to said connecting lines by a plurality of second data connections, said cells capable of being combined and facultatively grouped into functional parts by means of said first and second data connections, said cells connected to said input and output ports; a compiler capable of programming and configuring said cells and facultatively grouping said cells into functional parts such that various logic functions and data manipulations among said cells and said functional parts can be realized, said compiler capable of manipulating said cells and said functional parts during operation of said data flow processor such that portions of the data flow processor not being manipulated are not halted or impaired in their operation.~~

2. The data processing system according to Claim 1, further comprising a memory, wherein said memory is coupled with said compiler and said memory is adapted to specify the configuration of said cells.

3. The data processing system according to Claim 2, wherein said compiler manages a program sequence based on data and programs stored in separate memories, in the sense of a Harvard structure.

4. The data processing system according to Claim 3, wherein said compiler comprises logically and structurally identical cells.

5. The data processing system according to Claim 4, wherein said compiler is adapted to dynamically reconfigure said data flow processor during a program sequence during operation of said data processing system, without influencing the data to be processed.

6. The data processing system according to Claim 1, wherein said compiler manages a program sequence based on data and programs stored in separate memories, in the sense of a Harvard structure.

7. The data processing system according to Claim 6, wherein said compiler comprises logically and structurally identical cells.

8. The data processing system according to Claim 7, wherein said compiler is adapted to dynamically reconfigure said data flow processor during a program sequence during operation of said data processing system, without influencing the data to be processed.

9. The data processing system according to Claim 1, wherein said compiler is adapted to dynamically reconfigure said data flow processor during a program sequence during operation of said data processing system, without influencing the data to be processed.

10. The data processing system according to Claim 1, wherein said data flow processor is configured to be capable of coordinating a plurality of data in

put/output units and a plurality of memory units such that said data processing system operates as a highly complex and complete computer system.

11. The data processing system according to Claim 10, wherein some functions of said data input/output units are capable of being implemented on said data flow processor.

Sub B2 C2

12. A data processing system for manipulating data, comprising; an integrated circuit data flow processor, said data flow processor comprising a plurality of logically and structurally identical cells arranged orthogonal relative to one another, a plurality of connecting lines disposed between each said cells and a plurality of input and output ports, said cells connected to neighboring cells by a plurality of first data connections, said cells also connected to said connecting lines by a plurality of second data connections, said cells capable of being combined and facultatively grouped into functional parts by means of said first and second data connections, said cells connected to said input and output ports, said cells capable of being programmed and configured to be facultatively grouped into functional parts such that various logic functions and data manipulations among said cells and said functional parts can be realized, wherein a plurality of said data flow processors are coupled in cascade form.

13. The data processing system according to Claim 12, wherein said data flow processor is configured to be capable of coordinating a plurality of data input/output units and a plurality of memory units such that said data processing system operates as a highly complex and complete computer system.

~~14. The data processing system according to Claim 13, wherein some functions of said data input/output units are capable of being implemented on said data flow processor.~~

add a³>

ADD B⁵>

add C⁵>

Abstract

Data Processing System

A data processing system, wherein a data flow processor (DFP) integrated circuit chip is provided which comprises a plurality of orthogonally arranged homogeneously structured cells, each cell having a plurality of logically same and structurally identically arranged modules. The cells are combined and facultatively grouped using lines and columns and connected to the input/output ports of the DFP. A compiler programs and configures the cells, each by itself and facultatively grouped, such that random logic functions and/or linkages among the cells can be realized. The manipulation of the DFP configuration is performed during DFP operation such that modification of function parts (MACROs) of the DFP can take place without requiring other function parts to be deactivated or being impaired.

08/544435

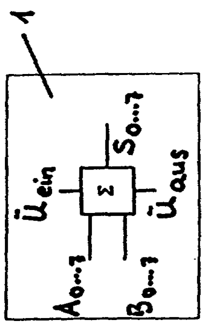


Fig.1

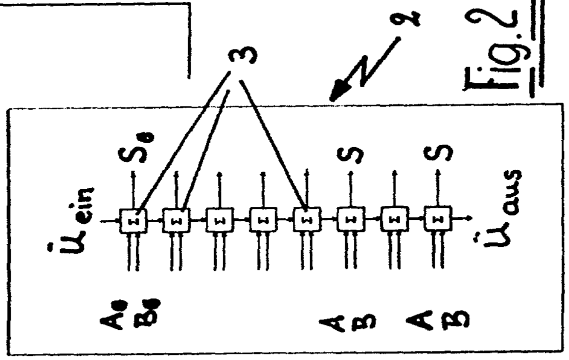


Fig.2

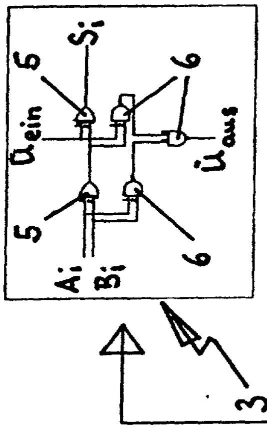


Fig.3

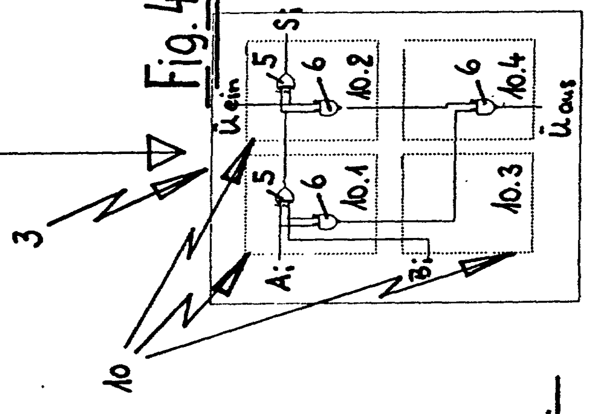


Fig.4

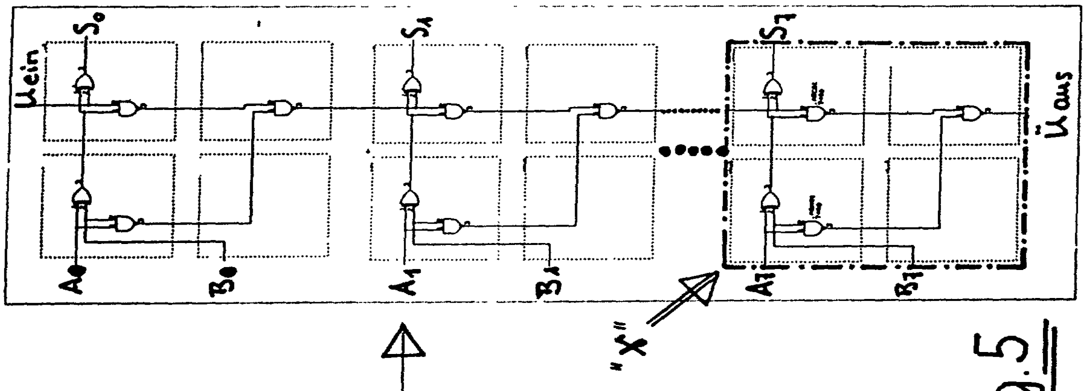


Fig.5

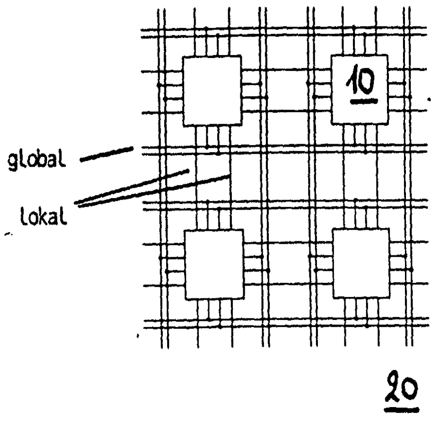


Fig.6

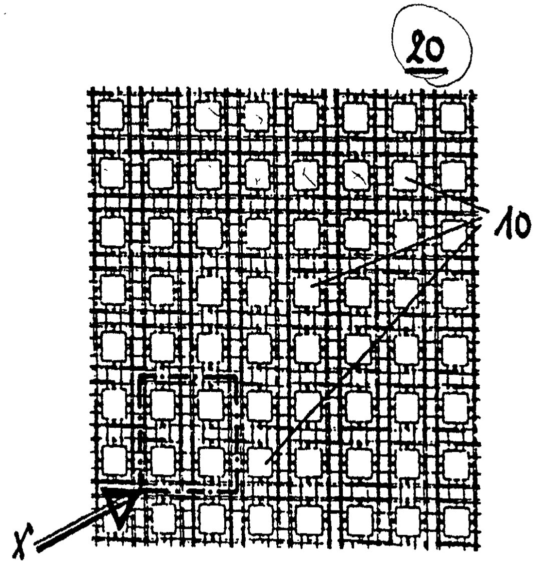


Fig.7

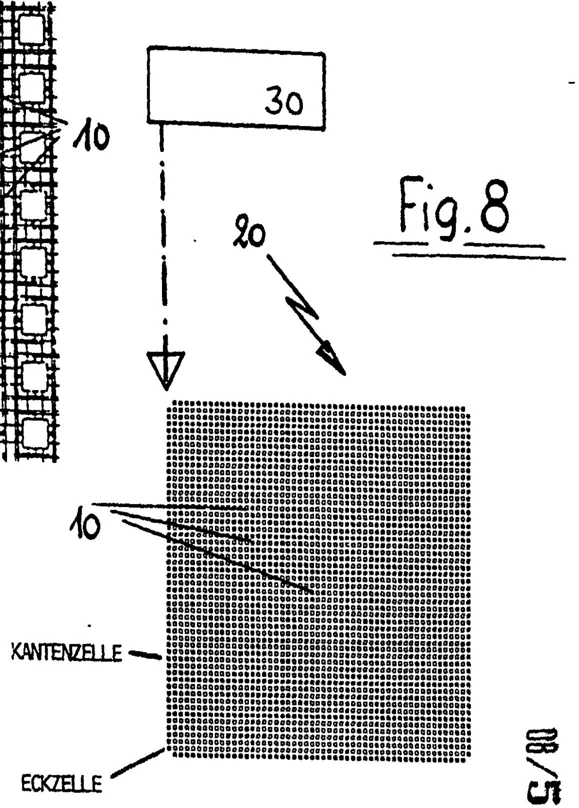


Fig.8

08/544435

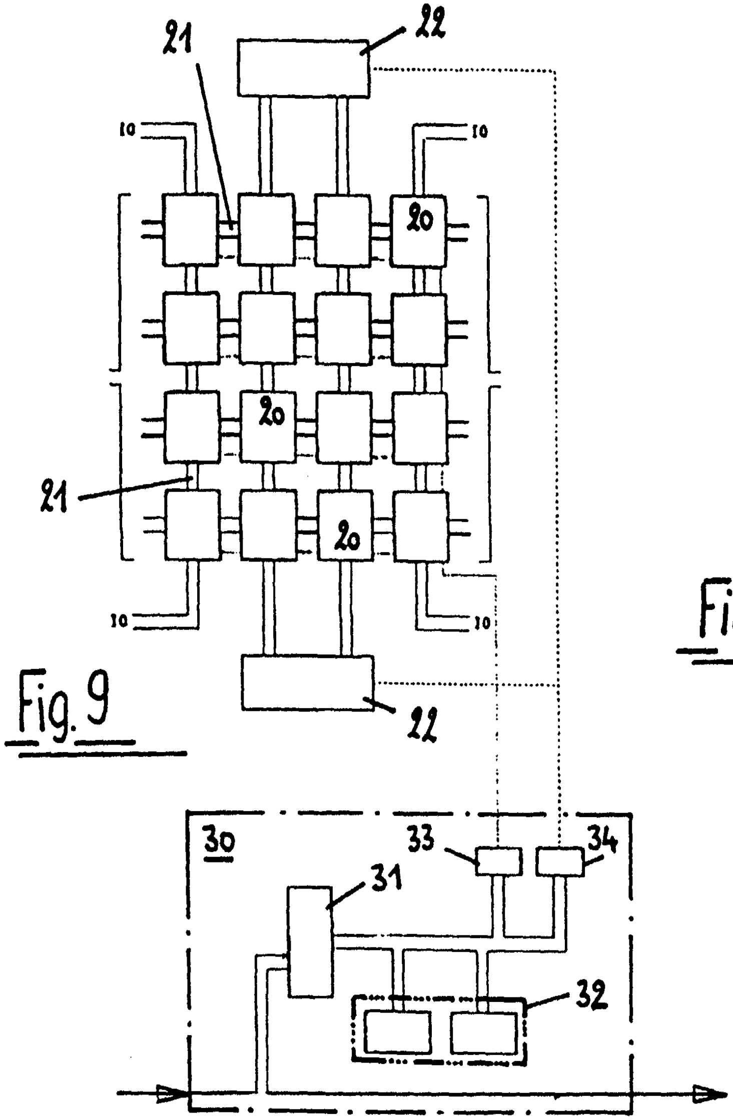


Fig. 9

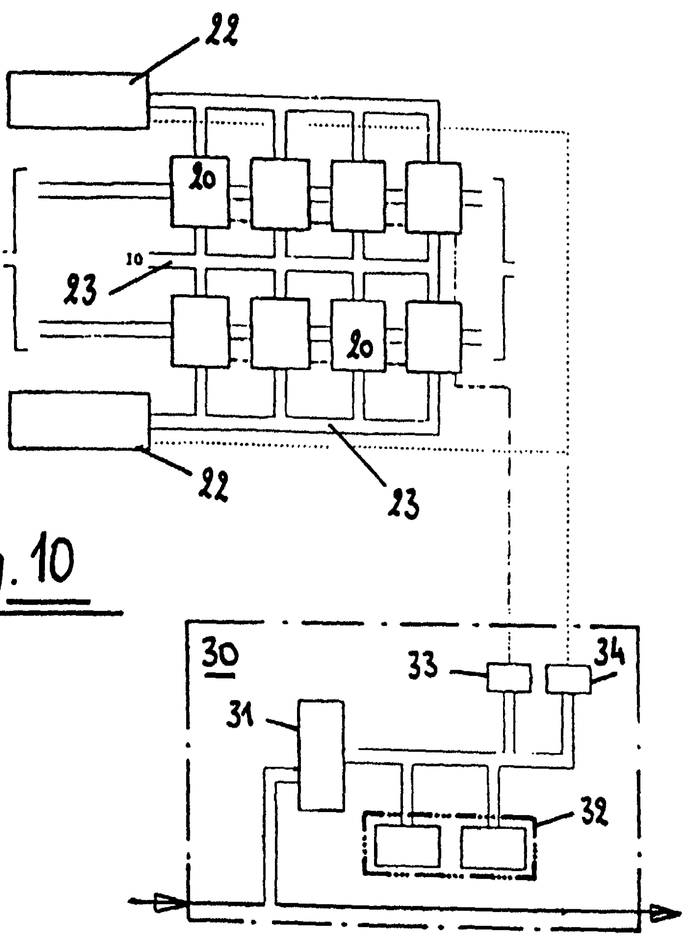


Fig. 10

08/544435

08/544435

Fig. 11 a

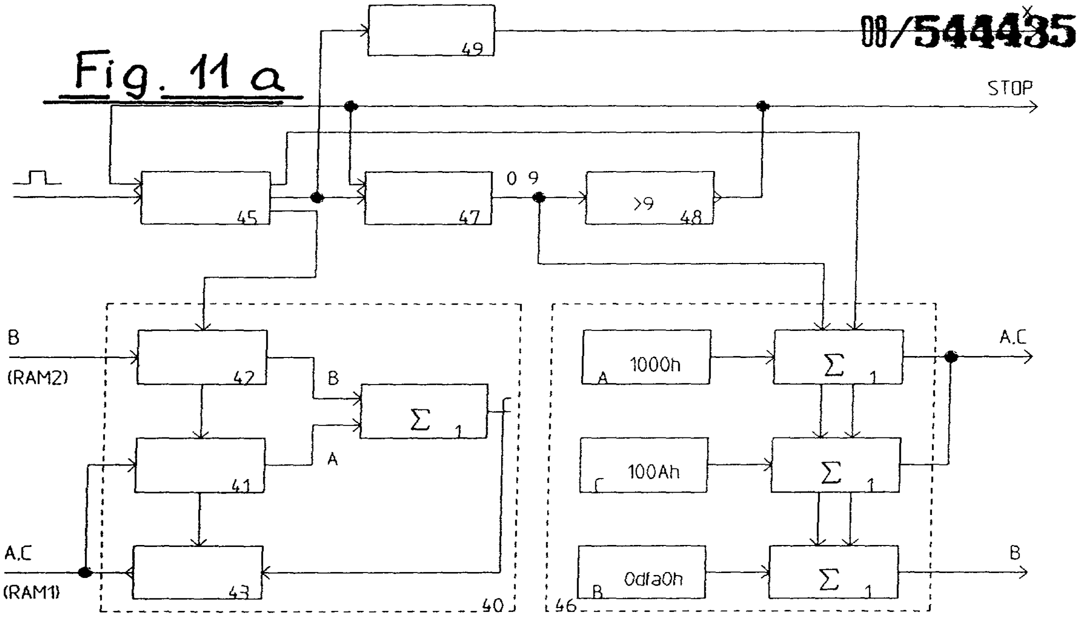
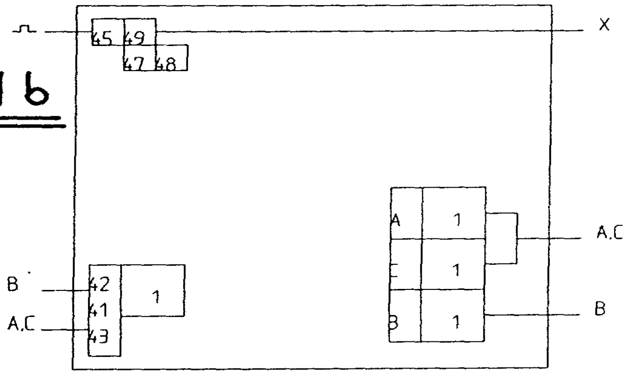


Fig. 11 b



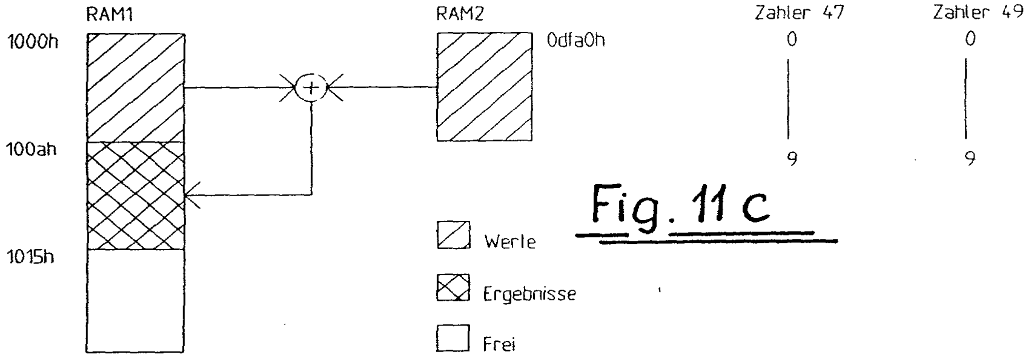


Fig. 11 c

VORB 08 4/11

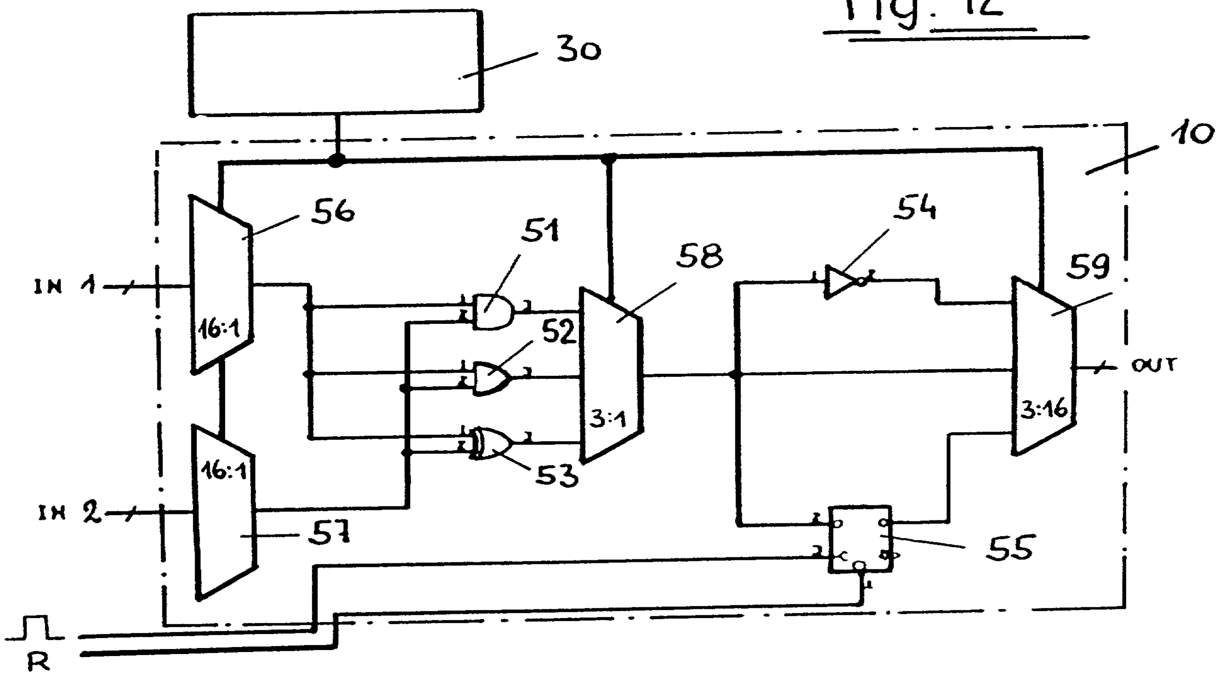

30
10
56
51
54
58
59
IN 1
16:1
52
OUT
3:1
3:16
53
16:1
IN 2
57
55
R

Fig. 12

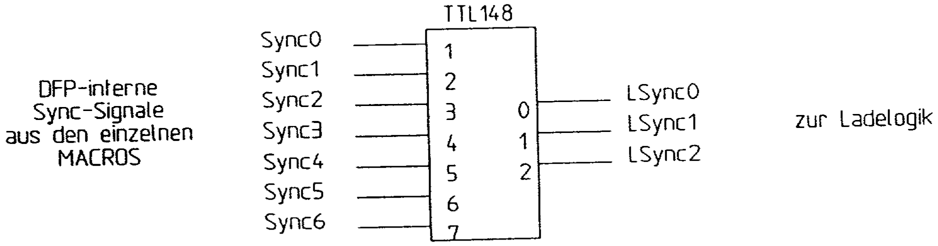

TTL148
DFP-interne
Sync-Signale
aus den einzelnen
MACROS
Sync0
Sync1
Sync2
Sync3
Sync4
Sync5
Sync6
1
2
3
4
5
6
7
0
1
2
LSync0
LSync1
LSync2
zur Ladelogik

Fig. 13

08/544435

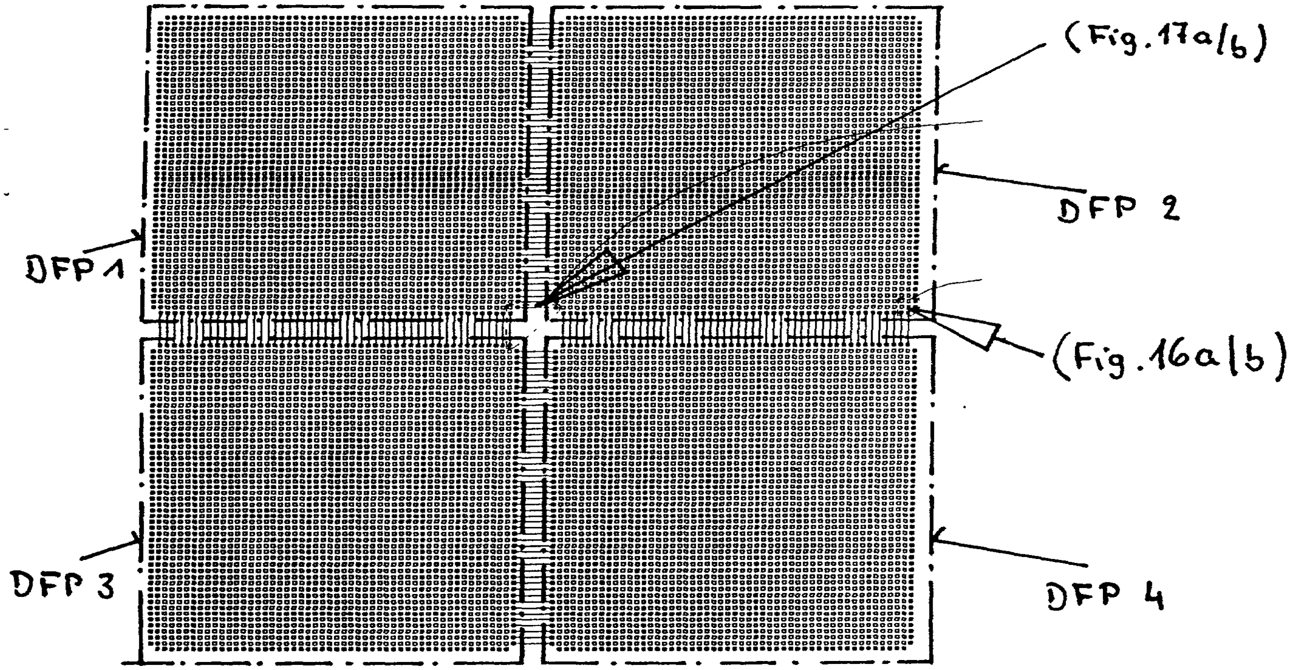


Fig. 14

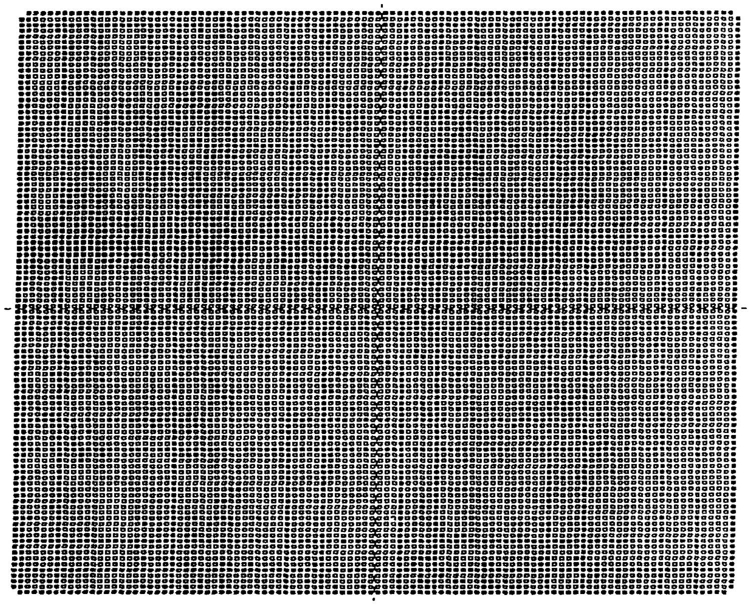


Fig. 15

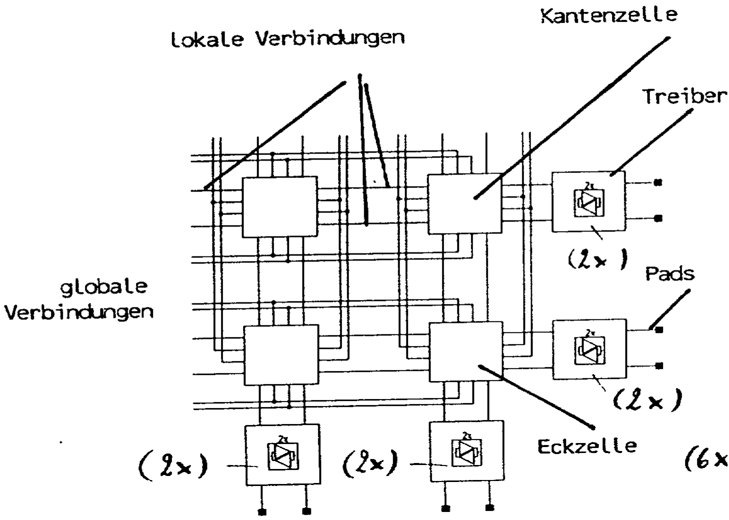


Fig. 16a

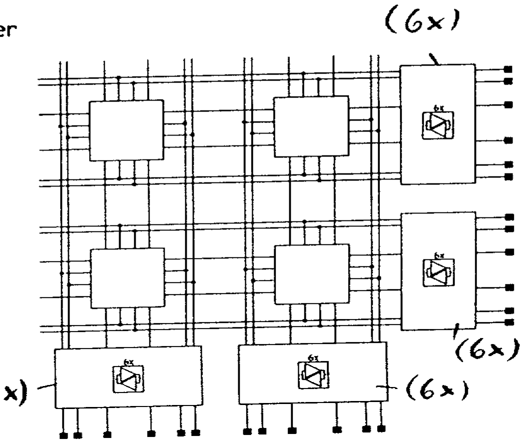


Fig. 16 b

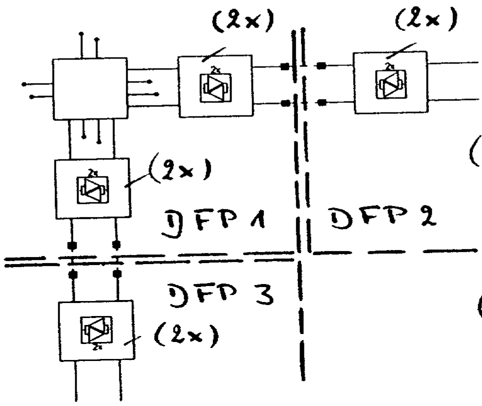


Fig. 17a

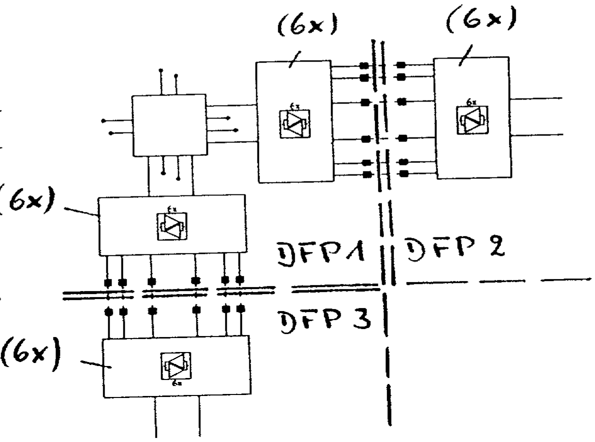


Fig. 17 b

08/544435

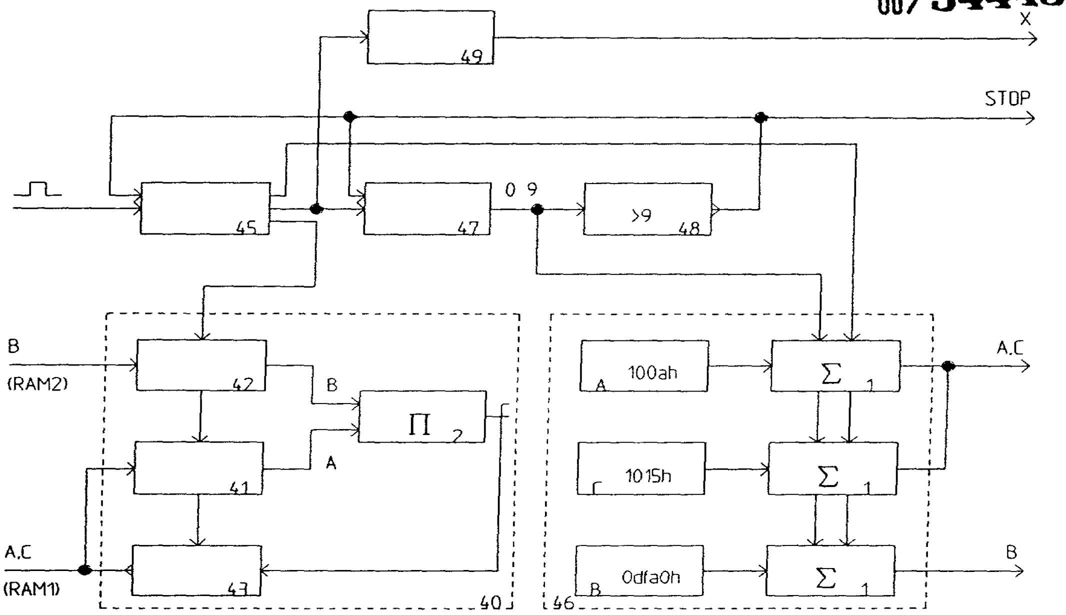


Fig. 18a

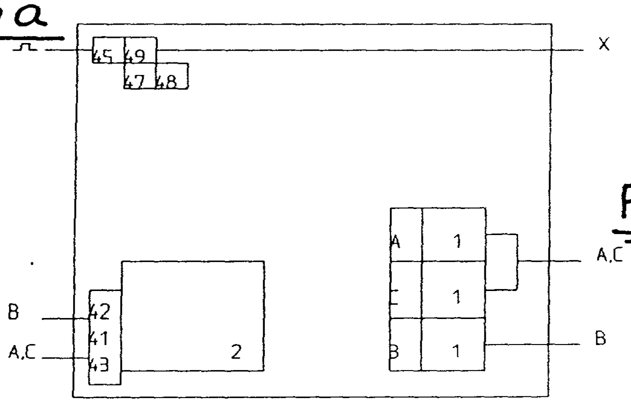


Fig. 18b

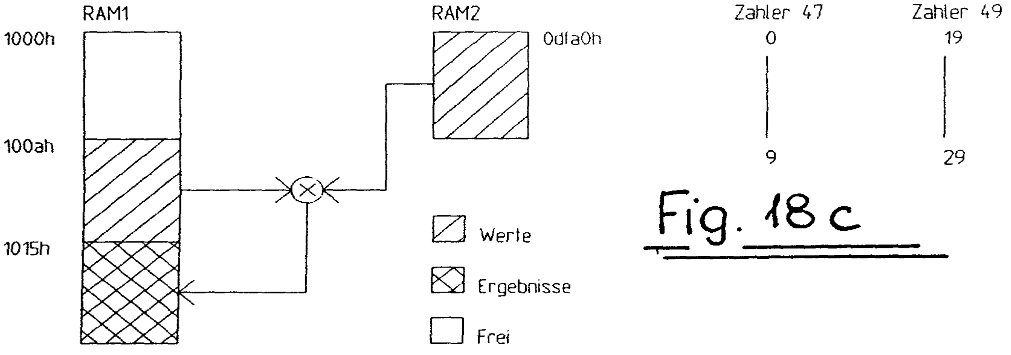
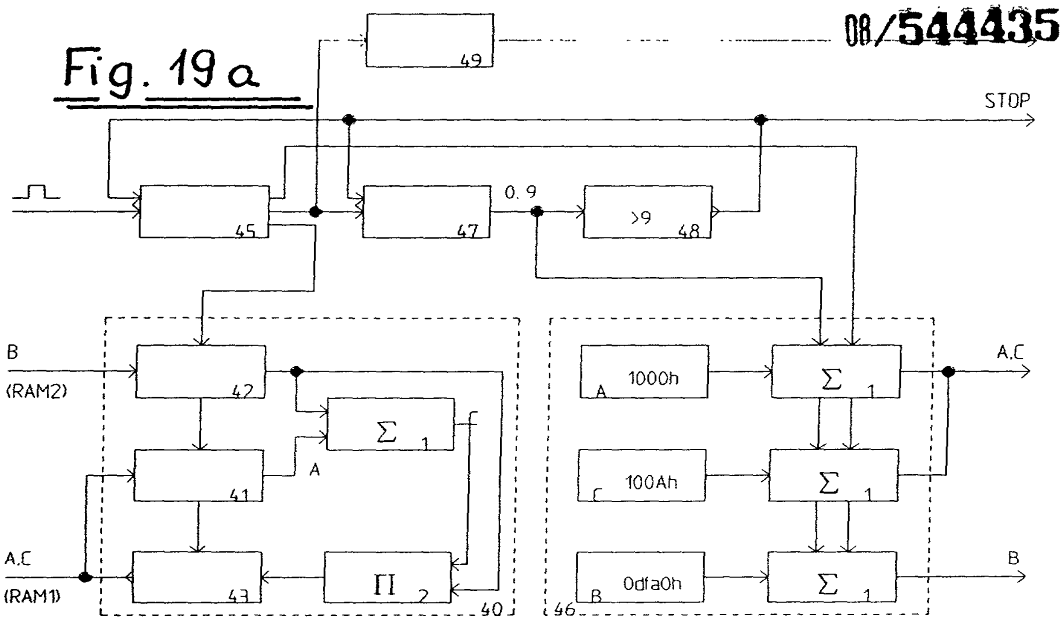


Fig. 18c

Fig. 19a



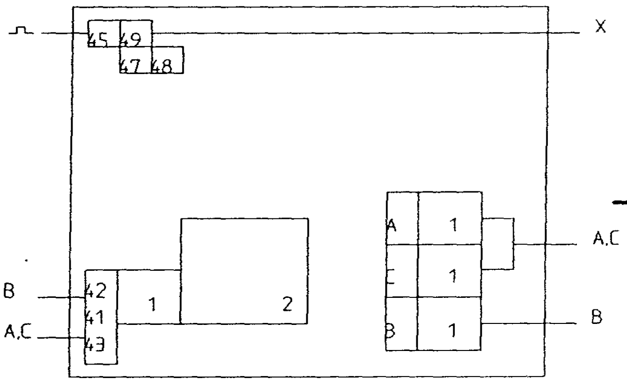


Fig. 19 b

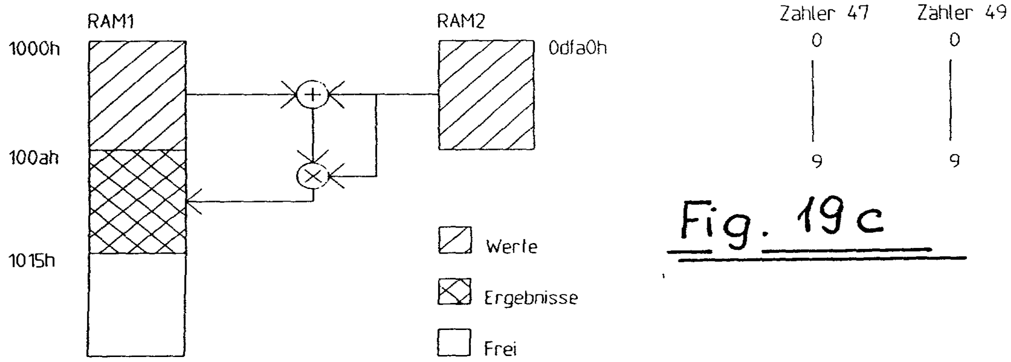


Fig. 19c

08/544435

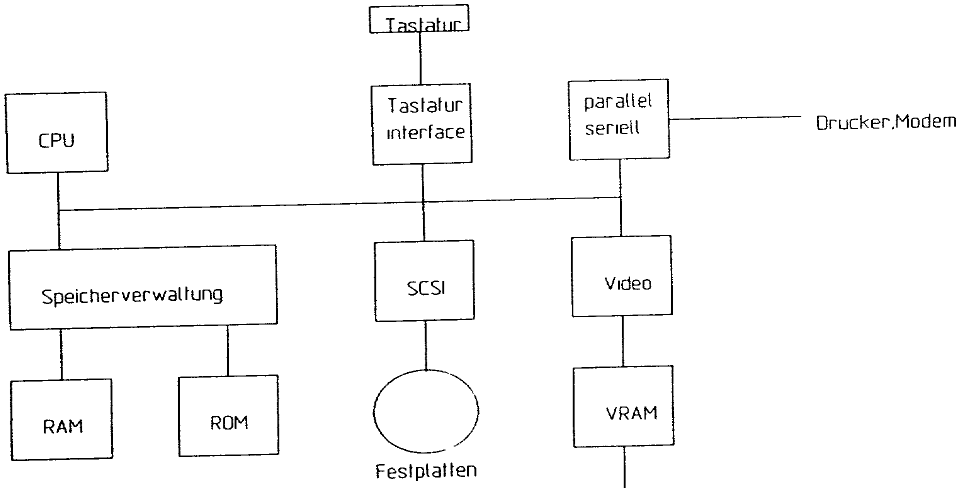


Fig. 20

Fig. 21

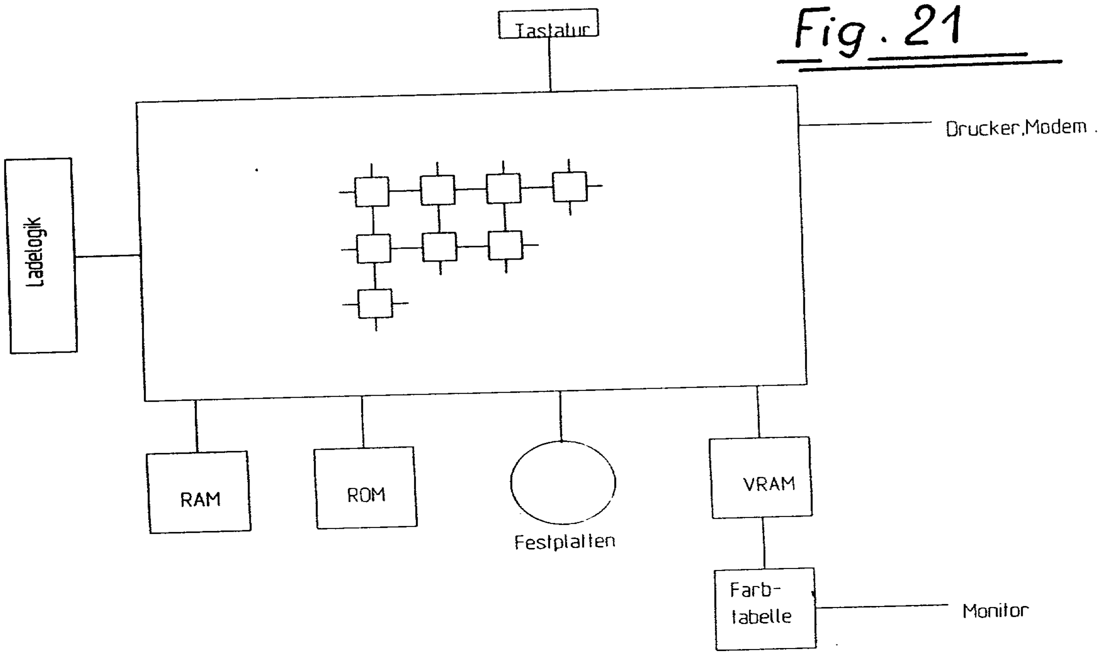
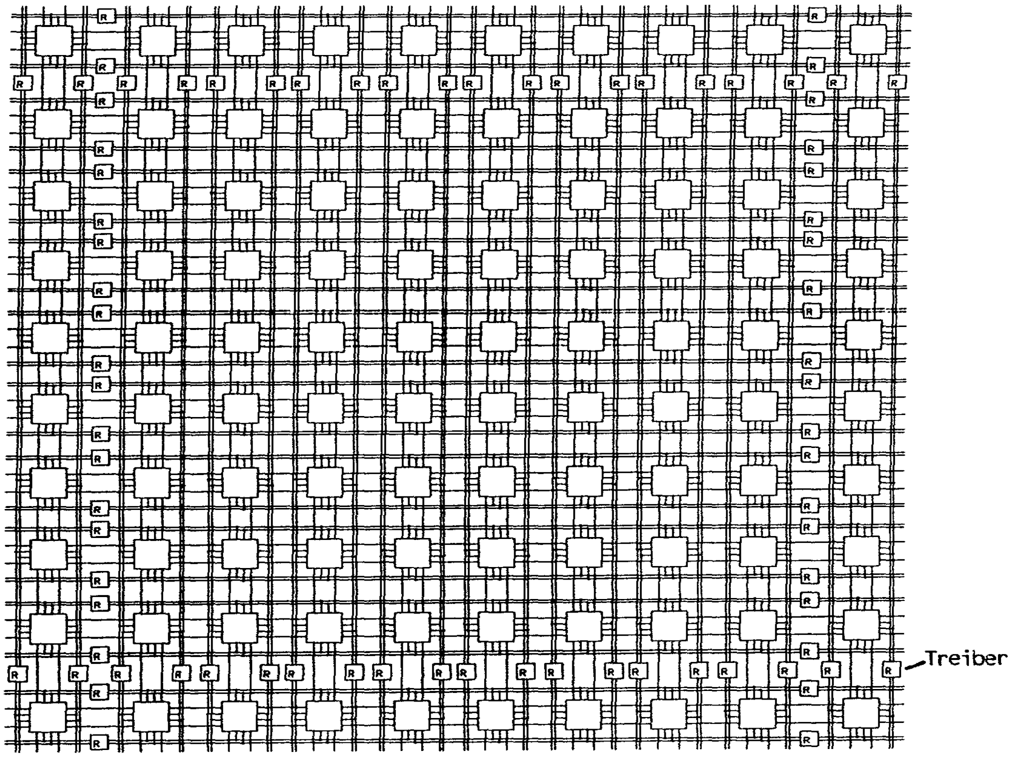


Fig. 22



#3

Declaration and Power of Attorney For Patent Application

Erklärung Für Patentanmeldungen Mit Vollmacht

German Language Declaration

Als nachstehend benannter Erfinder erkläre ich hiermit an Eides Statt:

dass mein Wohnsitz, *meine Postanschrift*, und meine Staatsangehörigkeit den im Nachstehenden nach meinem Namen aufgeführten Angaben entsprechen,

dass ich, nach bestem Wissen der ursprüngliche, erste und alleinige Erfinder (falls nachstehend nur ein Name angegeben ist) oder ein ursprünglicher, erster und Miterfinder (falls nachstehend mehrere Namen aufgeführt sind) des Gegenstandes bin, für den dieser Antrag gestellt *wird und für den* ein Patent beantragt wird für die Erfindung mit dem Titel:

____

____

________________________,

deren Beschreibung

(zutreffendes ankreuzen)

□ hier beigefügt ist.

□ am ________________ unter der

Anmeldungsseriennummer ________________

eingereicht wurde und am ________________,
abgeändert wurde (falls tatsächlich abgeändert).

Ich bestätige hiermit, dass ich den Inhalt der obigen Patentanmeldung einschliesslich der Ansprüche durchgesehen und verstanden habe, die eventuell durch einen Zusatzantrag wie oben erwähnt abgeändert wurde.

Ich erkenne meine Pflicht zur Offenbarung irgendwelcher Informationen, die für die Prüfung der vorliegenden Anmeldung in Einklang mit Absatz 37, Bundesgesetzbuch, Paragraph 1.56(a) von Wichtigkeit sind, an.

Ich beanspruche hiermit ausländische Prioritätsvorteile gemäss Abschnitt 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 119 aller unten angegebenen Auslandsanmeldungen für ein Patent oder eine Erfindersurkunde, und habe auch alle Auslandsanmeldungen für ein Patent oder eine Erfindersurkunde nachstehend gekennzeichnet, die ein Anmeldedatum haben, das vor dem Anmeldedatum der Anmeldung liegt, für die Priorität beansprucht wird.

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (*if plural names are listed below*) *of the subject matter which is claimed and for which a patent is sought on the invention entitled*

DATA PROCESSING SYSTEM

____

________________________,

the specification of which

(check one)

□ is attached hereto.

☒ was filed on 17 November 1995 as

Application Serial No. 08/544,435

and was amended on ________________
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

EL234414395US

Note
Paper No. 17
Declaration

German Language Declaratio

Prior foreign applications
Priorität beansprucht

			Priority Claimed	
DE 4316036.0	Germany	13 May 1993		
(Number) (Nummer)	(Country) (Land)	(Day/Month/Year Filed) (Tag/Monat/Jahr eingereicht)	□ Yes Ja	☒ No Nein
DE 4416881.0	Germany	13 May 1994		
(Number) (Nummer)	(Country) (Land)	(Day/Month/Year Filed) (Tag/Monat/Jahr eingereicht)	□ Yes Ja	☒ No Nein
(Number) (Nummer)	(Country) (Land)	(Day/Month/Year Filed) (Tag/Monat/Jahr eingereicht)	□ Yes Ja	□ No Nein

Ich beanspruche hiermit gemäss Absatz 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 120, den Vorzug aller unten aufgeführten Anmeldungen und falls der Gegenstand aus jedem Anspruch dieser Anmeldung nicht in einer früheren amerikanischen Patentanmeldung laut dem ersten Paragraphen des Absatzes 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 112 offenbart ist, erkenne ich gemäss Absatz 37, Bundesgesetzbuch, Paragraph 1.56(a) meine Pflicht zur Offenbarung von Informationen an, die zwischen dem Anmeldedatum der früheren Anmeldung und dem nationalen oder PCT internationalen Anmeldedatum dieser Anmeldung bekannt geworden sind.

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)
(Anmeldeseriennummer)

(Filing Date)
(Anmeldedatum)

(Status)
(patentiert, anhängig, aufgegeben)

(Status)
(patented, pending, abandoned)

(Application Serial No.)
(Anmeldeseriennummer)

(Filing Date)
(Anmeldedatum)

(Status)
(patentiert, anhängig, aufgegeben)

(Status)
(patented, pending, abandoned)

Ich erkläre hiermit, dass alle von mir in der vorliegenden Erklärung gemachten Angaben nach meinem besten Wissen und Gewissen der vollen Wahrheit entsprechen, und dass ich diese eidesstattliche Erklärung in Kenntnis dessen abgebe, dass wissentlich und vorsätzlich falsche Angaben gemäss Paragraph 1001, Absatz 18 der Zivilprozessordnung der Vereinigten Staaten von Amerika mit Geldstrafe belegt und/oder Gefängnis bestraft werden koennen, und dass derartig wissentlich und vorsätzlich falsche Angaben die Gültigkeit der vorliegenden Patentanmeldung oder eines darauf erteilten Patentes gefährden können.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

German Language Declaration

VERTRETUNGSVOLLMACHT: Als benannter Erfinder beauftrage ich hiermit den nachstehend benannten Patentanwalt (oder die nachstehend benannten Patentanwälte) und/oder Patent-Agenten mit der Verfolgung der vorliegenden Patentanmeldung sowie mit der Abwicklung aller damit verbundenen Geschäfte vor dem Patent-und Warenzeichenamt: *(Name und Registrationsnummer anführen)*

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (*list name and registration number*)

John F. Hoffman, Reg. No. 26,280; Anthony Niewyk, Reg. No. 24,871; Kevin R. Erdman, Reg. No. 33,687; Debra L. Schroeder, Reg. No. 37,717; Thomas A. Miller, Reg. No. P40,091; Brian C. Pauls, Reg. No. P40,122; Arthur R. Whale, Reg. No. 18,778; Lawrence A. Steward, Reg. No. 32,309; Edward J. Prein, Reg. No. 37,212; James D. Hall, Reg. No. 24,893; and Ken C. Decker, Reg. No. 25,422

Telefongespräche bitte richten an:
(Name und Telefonnummer)

Direct Telephone Calls to: (*name and telephone number*)
John F. Hoffman
219-424-8000

Postanschrift:

Send Correspondence to:
John F. Hoffman
BAKER & DANIELS
111 East WAyne Street
Fort Wayne, Indiana 46802

Voller Name des einzigen oder ursprünglichen Erfinders: 100	Full name of sole or first inventor Martin Vorbach
Unterschrift des Erfinders Datum	Inventor's signature Date Martin Vorbach 17.01.96
Wohnsitz	Residence Karlsruhe, Germany DEX
Staatsangehörigkeit	Citizenship Germany
Postanschrift	Post Office Address Hagebuttenweg 36, D-76149 Karlsruhe
	Germany
Voller Name des zweiten Miterfinders (falls zutreffend)	Full name of second joint inventor, if any
Unterschrift des Erfinders Datum	Second Inventor's signature Date
Wohnsitz	Residence
Staatsangehörigkeit	Citizenship
Postanschrift	Post Office Address

(Bitte entsprechende Informationen und Unterschriften im Falle von dritten und weiteren Miterfindern angeben).

(Supply similar information and signature for third and subsequent joint inventors.)